**PCT** WELTORGANISATION FÜR GEISTIGES EIGENTUM
Internationales Büro

INTERNATIONALE ANMELDUNG VERÖFFENTLICHT NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT)

(51) Internationale Patentklassifikation 6 : H04L 12/00

A2

(11) Internationale Veröffentlichungsnummer: **WO 99/08419**

(43) Internationales Veröffentlichungsdatum: 18. Februar 1999 (18.02.99)

(21) Internationales Aktenzeichen: PCT/DE98/02265

(22) Internationales Anmeldedatum: 6. August 1998 (06.08.98)

(30) Prioritätsdaten:
197 34 229.9 7. August 1997 (07.08.97) DE

(71) Anmelder *(für alle Bestimmungsstaaten ausser US)*: SIEMENS AKTIENGESELLSCHAFT [DE/DE]; Wittelsbacherplatz 2, D–80333 München (DE).

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: SCHÄFER, Ralf [DE/DE]; Dürrenmettstetter Strasse 32, D–72175 Dornhan (DE). GITSELS, Martin [DE/DE]; Valleystrasse 32, D–81371 München (DE).

(74) Gemeinsamer Vertreter: SIEMENS AKTIENGESELLSCHAFT; Postfach 22 16 34, D–80506 München (DE).

(81) Bestimmungsstaaten: CN, JP, US, europäisches Patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

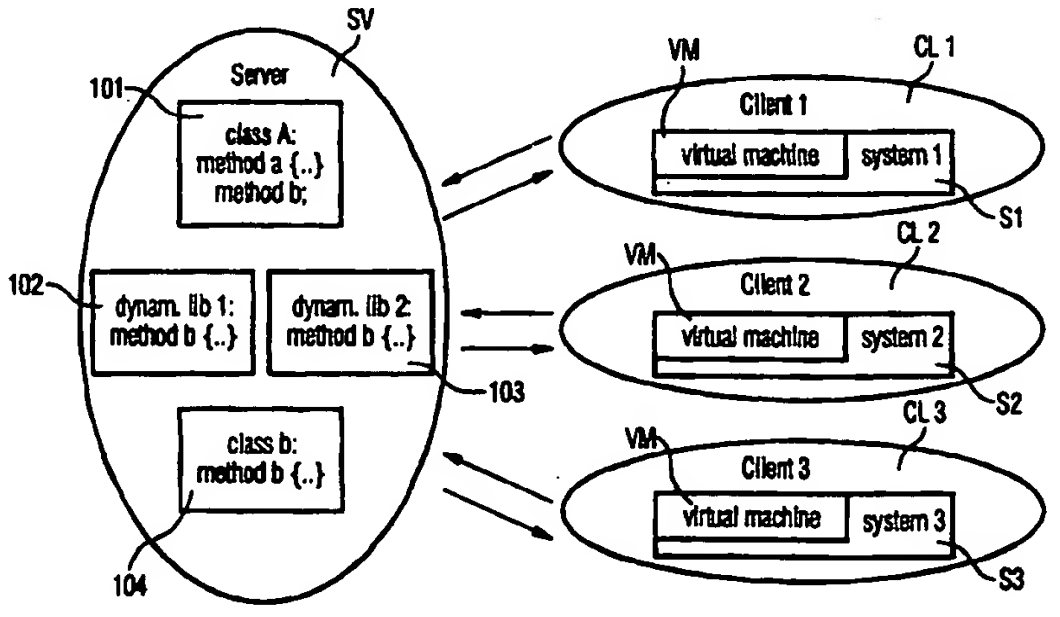
**Veröffentlicht**
*Ohne internationalen Recherchenbericht und erneut zu veröffentlichen nach Erhalt des Berichts.*

(54) Title: METHOD FOR LOADING A FUNCTION PROVIDED BY A FIRST COMPUTER (SERVER) ONTO A SECOND COMPUTER (CLIENT)

(54) Bezeichnung: VERFAHREN ZUM LADEN EINER FUNKTION, DIE VON EINEM ERSTEN RECHNER (SERVER) BEREITGESTELLT WIRD, AUF EINEN ZWEITEN RECHNER (CLIENT)

**(57) Abstract**

A function is to be loaded from a first computer onto a second computer via a network, preferably a computer network or communications network. The second computer contains, in addition to the operating system, a platform–independent system, preferably a JAVA virtual machine which ensures that the platform–independent code (JAVA byte code) from the operating system of the second computer can run on the second computer. In addition to the platform–independent code, a platform–dependent code, particular to the hardware of the second computer, can be loaded from the first computer, thus avoiding the operation time losses that result from the platform–independent code. To this end, the first computer comprises the necessary platform–dependent functions, preferably as dynamic libraries, which are optionally requested by the second computer, so as to obtain operation time gains on the second computer.



**(57) Zusammenfassung**

Von einem ersten Rechner soll über ein Netz, vorzugsweise ein Rechnernetz oder ein Kommunikationsnetz, eine Funktion auf einen zweiten Rechner geladen werden. Der zweite Rechner enthält dazu neben einem Betriebssystem ein plattformunabhängiges System, vorzugsweise eine JAVA-Virtual-Maschine, das gewährleistet, daß von dem Betriebssystem des zweiten Rechners plattformunabhängiger Code (JAVA-Bytecode) auf diesem zweiten Rechner ablaufen kann. Neben plattformunabhängigem Code kann auch plattformabhängiger Code, speziell für die Hardware des zweiten Rechners, von dem ersten Rechner geladen werden, wobei die durch plattformunabhängigen Code resultierenden Laufzeiteinbußen vermieden werden. Dazu umfaßt der erste Rechner die benötigten plattformabhängigen Funktionen, vorzugsweise als dynamische Bibliotheken, die ggf. von dem zweiten Rechner angefordert werden, um Laufzeitgewinne auf dem zweiten Rechner zu erzielen.

***LEDIGLICH ZUR INFORMATION***

Codes zur Identifizierung von PCT-Vertragsstaaten auf den Kopfbögen der Schriften, die internationale Anmeldungen gemäss dem PCT veröffentlichen.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| AL | Albanien | ES | Spanien | LS | Lesotho | SI | Slowenien |
| AM | Armenien | FI | Finnland | LT | Litauen | SK | Slowakei |
| AT | Österreich | FR | Frankreich | LU | Luxemburg | SN | Senegal |
| AU | Australien | GA | Gabun | LV | Lettland | SZ | Swasiland |
| AZ | Aserbaidschan | GB | Vereinigtes Königreich | MC | Monaco | TD | Tschad |
| BA | Bosnien-Herzegowina | GE | Georgien | MD | Republik Moldau | TG | Togo |
| BB | Barbados | GH | Ghana | MG | Madagaskar | TJ | Tadschikistan |
| BE | Belgien | GN | Guinea | MK | Die ehemalige jugoslawische Republik Mazedonien | TM | Turkmenistan |
| BF | Burkina Faso | GR | Griechenland | | | TR | Türkei |
| BG | Bulgarien | HU | Ungarn | ML | Mali | TT | Trinidad und Tobago |
| BJ | Benin | IE | Irland | MN | Mongolei | UA | Ukraine |
| BR | Brasilien | IL | Israel | MR | Mauretanien | UG | Uganda |
| BY | Belarus | IS | Island | MW | Malawi | US | Vereinigte Staaten von Amerika |
| CA | Kanada | IT | Italien | MX | Mexiko | | |
| CF | Zentralafrikanische Republik | JP | Japan | NE | Niger | UZ | Usbekistan |
| CG | Kongo | KE | Kenia | NL | Niederlande | VN | Vietnam |
| CH | Schweiz | KG | Kirgisistan | NO | Norwegen | YU | Jugoslawien |
| CI | Côte d'Ivoire | KP | Demokratische Volksrepublik Korea | NZ | Neuseeland | ZW | Zimbabwe |
| CM | Kamerun | | | PL | Polen | | |
| CN | China | KR | Republik Korea | PT | Portugal | | |
| CU | Kuba | KZ | Kasachstan | RO | Rumänien | | |
| CZ | Tschechische Republik | LC | St. Lucia | RU | Russische Föderation | | |
| DE | Deutschland | LI | Liechtenstein | SD | Sudan | | |
| DK | Dänemark | LK | Sri Lanka | SE | Schweden | | |
| EE | Estland | LR | Liberia | SG | Singapur | | |

Beschreibung

**Verfahren zum Laden einer Funktion, die von einem ersten Rechner (Server) bereitgestellt wird, auf einen zweiten Rechner (Client)**

Die Erfindung betrifft ein Verfahren zum Laden einer Funktion, die von einem ersten Rechner (Server) bereitgestellt wird, auf einen zweiten Rechner (Client), wobei der erste Rechner und der zweite Rechner über ein Netz verbunden sind.

Ein Netz bezeichnet ein Medium, über das Rechner miteinander verbunden sind und über das Nachrichten zwischen den Rechner, sei es verbindungslos oder verbindungsorientiert, ausgetauscht werden. Beispiele für solche Netze sind Rechnernetze und Kommunikationsnetze.

Mit Funktion wird eine Komponente bezeichnet, die auf einem zweiten Rechner benötigt und auf einem ersten Rechner zur Verfügung gestellt wird. Derartige Funktionen können beispielsweise Programme, die über das Netz von dem ersten Rechner geladen werden sollen, Prozeduren im Verständnis einer strukturierten Programmiersprache oder Methoden im objektorientierten Sinn sein.

Die Programmiersprache JAVA ist dem Fachmann hinlänglich bekannt. Ein wichtiges Merkmal von JAVA ist seine Unabhängigkeit von einer bestimmten Plattform (plattformunabhängig). JAVA läuft auf jedem System, auf dem eine JAVA-Virtual-Machine implementiert ist. Diese JAVA-Virtual-Machine übernimmt die Ausführung des JAVA-Programms, wobei eine kontrollierte Übertragung mit anschließender Integritätsüberprüfung des geladenen Codes erfolgen kann. Ein JAVA-Applet bezeichnet eine in JAVA geschriebene Applikation, die auf Netzbrowsern ausgeführt wird und somit auf der in dem

Netzbrowser integrierten JAVA-Virtual-Machine läuft. Allgemein ist jedes JAVA-Applet auch ein JAVA-Programm.

Da, wie erwähnt, für ein ablauffähiges JAVA-Programm (z.B. ein JAVA-Applet) eine JAVA-Virtual-Machine notwendig ist, sind deutliche Laufzeiteinbußen im Vergleich zu einem eigens für den jeweiligen Rechner geschriebenen Programm (plattformabhängig) hinzunehmen. JAVA-Bytecode bezeichnet einen Programmcode, der plattformunabhängig auf jedem Rechner, der über eine JAVA-Virtual-Machine verfügt, ablaufen kann. Der JAVA-Bytecode ist in JAVA-Class-Files integriert, die weitere für die JAVA-Virtual-Machine nützliche Information bereitstellen, insbesondere eine Semantik für die Umgebung der JAVA-Virtual-Machine beinhalten.

Demzufolge kann unterschieden werden in plattformabhängige, d.h. für den jeweiligen Rechner spezifizierte, und plattformunabhängige, d.h. unabhängig von dem speziellen Rechner lauffähige, Programme. Mittels eines plattformabhängigen Programms ist der Einsatz von Programmcode (auch: Code) möglich, der spezielle Hardwaremerkmale des jeweiligen Rechners ausnutzt. Damit ist ein signifikanter Laufzeitgewinn bei Programmausführung verbunden.

In existierenden Systemen, die verschiedene Rechner miteinander über plattformunabhängige Programme (z.B. JAVA) verbinden, ist es von Nachteil, daß ein plattformunabhängiges Programm deutliche Laufzeiteinbußen mit sich bringt, wohingegen bei auf spezielle Hardware, insbesondere Multimedia-Hardware, abgestimmter Programmcode immense Laufzeitvorteile mit sich brächte.

Die Aufgabe der Erfindung besteht darin, ein Verfahren zum Laden einer Funktion von einem ersten Rechner auf einen zweiten Rechner anzugeben, wobei die oben erwähnten Nachteile vermieden werden.

Diese Aufgabe wird gemäß den Merkmalen des Patentanspruchs 1 gelöst.

Das erfindungsgemäße Verfahren ermöglicht das Laden einer Funktion, die von einem ersten Rechner (Server) bereitgestellt wird, auf einen zweiten Rechner (Client), wobei der erste Rechner und der zweite Rechner über ein Netz verbunden sind. Dazu wird von dem zweiten Rechner über das Netz bei dem ersten Rechner angefragt, ob die Funktion in einem plattformabhängigen Code verfügbar ist. Falls ein plattformabhängiger Code nicht verfügbar ist, wird ein plattformunabhängiger Code der Funktion von dem ersten Rechner geladen und das Verfahren wird beendet. Ansonsten (der plattformabhängige Code ist verfügbar) wird zunächst überprüft, ob dieser plattformabhängige Code der Funktion schon auf dem zweiten Rechner vorhanden ist und, so dies der Fall ist, wird dieser plattformabhängige Code geladen und das Verfahren beendet. Ist hingegen der plattformabhängige Code auf dem zweiten Rechner nicht vorhanden, so wird mittels einer Eingabe entschieden, ob der plattformabhängige Code vom ersten Rechner geladen werden soll. Ergibt sich aus der Eingabe, daß der plattformabhängige Code nicht geladen werden soll, wird der plattformunabhängige Code geladen und das Verfahren beendet. Wird mittels der Eingabe entschieden, daß der plattformabhängige Code geladen werden soll, so wird nach einer Verhandlung zwischen dem ersten Rechner und dem zweiten Rechner der plattformabhängige Code der Funktion auf den zweiten Rechner geladen.

Eine Weiterbildung der Erfindung besteht darin, daß die Verhandlung folgende Schritte umfaßt:
Der zweite Rechner fordert von dem ersten Rechner eine Liste aller verfügbaren plattformabhängigen Realisierungen der Funktion an. Diese Liste wird von dem ersten Rechner an den zweiten Rechner übertragen. Der zweite Rechner wählt eine geeignete Realisierung aus, teilt dies dem ersten Rechner mit

und fordert eine Übertragung dieser Realisierung der Funktion an. Dadurch wird sichergestellt, daß eine passende plattformabhängige Realisierung zu dem zweiten Rechner übertragen wird. Hierbei sei angemerkt, daß eine solche plattformabhängige Realisierung insbesondere ein für die jeweilige Plattform erstelltes Programm oder ein für die jeweilige Plattform kompilierter Object-Code sein kann.

Eine andere Weiterbildung besteht darin, daß nach dem Laden plattformabhängigen Codes von dem ersten Rechner auf den zweiten Rechner eine Sicherheitsüberprüfung dieses plattformabhängigen Codes durchgeführt wird. Dabei kann diese Sicherheitsüberprüfung dadurch erfolgen, daß, ehe der geladene plattformabhängige Code auf dem zweiten Rechner ausgeführt wird, eine Integritätsprüfung dieses Codes erfolgt, vorzugsweise mittels eines Virensuchprogramms.

Im Rahmen einer zusätzlichen Weiterbildung ist es möglich die Eingabe auf eine der folgenden Arten durchzuführen:

a) Ein Benutzer gibt an, ob er plattformabhängigen Code von dem ersten Rechner laden will. Hierbei kann der Benutzer die Entscheidung treffen und das Risiko eingehen, korrupten, also fehlerhaften oder mit beispielsweise. Viren behafteten, Code von dem ersten Rechner zu empfangen.

b) Die Eingabe erfolgt automatisiert dadurch, daß eine Liste auf dem zweiten Rechner gehalten wird, die Kennungen von Rechnern enthält, die in einer sicheren Umgebung stehen (Trusted Servers).

c) Die Eingabe ist dauerhaft auf *"Laden von plattformabhängigen Code"* eingestellt, wenn der erste Rechner keine sicherheitsrelevanten Daten und/oder Programme enthält. Dies ist vorzugsweise bei speziell dafür vorgesehenen Netzwerkcomputern der Fall. Generell

ist von Vorteil, wenn die geladenen Funktionen nur für die zeitlich begrenzte Sitzung benötigt werden, der zweite Rechner nach der Sitzung ausgeschaltet wird und/oder die über das Netz erhaltenen Daten nicht persistent abspeichert.

Auch ist es eine Weiterbildung, daß es sich bei dem plattformunabhängigen Code um JAVA-Bytecode, der in JAVA-Class-Files integriert ist, handelt.

Weiterbildungen der Erfindung ergeben sich auch aus den abhängigen Ansprüchen.

Anhand der folgenden Figuren werden Ausführungsbeispiele der Erfindung näher dargestellt.

Es zeigen

Fig.1 ein System aus verteilten Rechnern mit plattform-abhängigen und plattformunabhängigen Funktionen,

Fig.2 ein Blockdiagramm für ein Verfahren zum Laden einer Funktion in plattformabhängigen oder plattformunabhängigem Code,

Fig.3 ein Ablaufdiagramm für eine Verhandlung darüber, welche Realisierung einer Funktion in plattform-abhängigem Code geladen werden soll.

Die in Fig.1 dargestellte Konfiguration stellt ein Client-/Server-System dar. Die Rechner CL1, CL2 und CL3 (Clients) wollen eine, vorzugsweise in JAVA geschriebene, Applikation auf deren Virtual-Machine VM ausführen. Die drei dargestellten Rechner CL1, CL2 und CL3 verfügen jeweils über unterschiedliche Betriebssysteme S1, S2 und S3, wobei auf jedem dieser Betriebssysteme eine Virtual-Machine VM läuft, die ihrerseits sicherstellt, daß plattformunabhängige JAVA-Class-Files, der für diese Virtual-Machine VM bestimmt ist, auf jedem der Rechner CL1, CL2 und CL3 ausgeführt werden kann. Handelt es sich bei den JAVA-Programmen um JAVA-

Applets, so ist ein Netzbrowser aktiv. Auf dem Server SV ist eine Klasse A mit einer plattformunabhängigen Implementierung einer Methode a enthalten (siehe Block 101).

Hierbei sein angemerkt, daß der Begriff "Methode" im Sinne der objektorientierten Programmierung verwendet wird, womit eine Prozedur oder Funktion bezeichnet ist, die mit einer Klasse assoziiert ist und als Reaktion auf eine Botschaft aufgerufen wird. Dabei ist eine Botschaft eine Aufforderung an ein Objekt, eine seiner Methoden auszuführen. Weitergehende Definitionen und Erläuterungen zur objektorientierten Programmierung mit den dort bevorzugten Begriffen sind dem Fachmann hinlänglich bekannt.

Die Implementation der Methode a in plattformunabhängigem Code, vorzugsweise in JAVA-Bytecode und nicht in plattformabhängigem Code bedeutet einen plattformübergreifenden Einsatz der Methode a in allen Rechnern CL1, CL2 und CL3 mit einer entsprechenden Virtual-Machine VM, wobei mit Verwendung der Virtual-Machine VM Laufzeiteinbußen in Kauf genommen werden.

Eine zweite Methode b ist innerhalb der Klasse A 101 lediglich deklariert; definiert wird die Methode b außerhalb der Klasse A auf drei verschiedene Arten:

1. In Form einer plattformabhängigen Bibliothek 102, die für das Betriebssystem S1 bestimmt ist;

2. als plattformabhängige Bibliothek 103, die für das Betriebssystem S2 bestimmt ist;

3. als eine plattformunabhängige Methode innerhalb einer Klasse b (siehe Block 104).

Fordert der Rechner CL1 Code für die Methode b an, so lädt er bisher die Klasse b und führt den verhältnismäßig langsamen

Bytecode aus, auch wenn der Rechner CL1 zur Durchführung der der Methode b zugrundeliegenden Funktionalität vorzugsweise über spezielle Multimedia-Hardware verfügt.

Um es zu ermöglichen, speziellen plattformabhängigen und auf Laufzeit optimierten Code in dem jeweiligen Rechner (hier der Rechner CL1) nutzen zu können, wird die Methode b in der Klasse A als "native" deklariert. Somit erfährt der Rechner CL1, daß die Methode b auch in plattformabhängigem Code auf dem Server SV verfügbar ist. Der Rechner CL1 lädt eine plattformabhängige Repräsentation der Methode b, falls diese speziell für seine Plattform, das Betriebssystem S1, verfügbar ist, von dem Server SV (im Beispiel von Fig.1 entspricht dieser Repräsentation der Block 102). Damit erhält der Rechner CL1 eine speziell auf die Hardware dieses Rechners CL1 abgestimmte und weniger laufzeitintensive Version der Methode b. Vorzugsweise führt der Rechner CL1 über eine vorgegebene Schnittstelle (JAVA-Native-Interface) die Methode b in der JAVA-Umgebung aus.

Die Anfrage eines Rechners CL1, CL2 oder CL3 beim Server SV, ob ein plattformabhängiger Code auf dem Server SV verfügbar sei und ob dieser ggf. auch auf den jeweiligen Rechner CL1, CL2 oder CL3 geladen werden soll, ist detailliert in Fig.2 dargestellt.

In einem Schritt 201 fragt der Rechner CL1 an, ob plattformabhängiger Code auf dem Server SV verfügbar sei. Ist dies nicht der Fall, wird plattformunabhängiger Code geladen (siehe Schritt 202 bzw. Block 104 in Fig.1 für eine plattformunabhängige Repräsentation der Methode b). Ist hingegen plattformabhängiger Code auf dem Server SV verfügbar, so wird in einem Schritt 203 festgestellt, ob auf dem jeweiligen Rechner CL1 der entsprechende plattformabhängige Code schon lokal verfügbar ist. In solch einem Fall wird in einem Schritt 204 auf den lokal verfügbaren Code zugegriffen. Dies kann beispielsweise durch

entsprechende Zwischenspeicherung (Cache-Speicher) gewährleistet werden.

Ist plattformabhängiger Code auf dem Server SV verfügbar, aber lokal auf dem Rechner CL1 nicht verfügbar, so wird in einem Schritt 205 entschieden, ob plattformabhängiger Code von dem Server SV auf den Rechner CL1 geladen werden soll.

Als Entscheidungshilfe dient eine Eingabe 206, durch die der Benutzer festlegen kann, ob er plattformabhängigen Code von speziell diesem Server SV laden will. Eine andere mögliche Eingabe ist eine Liste, die Kennzeichnungen von Servern enthält, die als vertrauenswürdig (Trusted Servers) gelten. Von solchen Servern kann ohne Sicherheitsrisiko plattformabhängiger Code geladen werden. Eine weitere Möglichkeit für eine Eingabe ist es, immer plattformabhängigen Code von einem Server zu laden. Diese Strategie eignet sich v.a. dann, wenn der Rechner CL1 ein reiner Netzwerkrechner ist und/oder keine sicherheitsrelevanten Daten und/oder Programme enthält.

Wird davon ausgegangen, daß aufgrund der Eingabe 206 kein plattformabhängiger Code von dem Server SV geladen werden soll, so wird in einem Schritt 207 plattformunabhängiger Code (siehe wieder Bock 104 in Fig.1) geladen. Soll hingegen plattformabhängiger Code geladen werden, so wird in einem Schritt 208 eine Verhandlung über die Art des zu ladenden Codes (siehe auch Beschreibung zu Fig.3) durchgeführt und in einem Schritt 209 der plattformabhängige Code geladen. Plattformabhängiger Code für den Rechner CL1 ist in Fig.1 durch den Block 102 angedeutet. Bevor der plattformabhängige Code auf dem Rechner CL1 ausgeführt wird, wird vorzugsweise eine Sicherheitsüberprüfung dieses neu geladenen Codes durchgeführt. Dies geschieht vorzugsweise mit speziellen Programmen, die die Integrität des geladenen Codes sicherstellen, vorzugsweise mittels eines Virensuchprogramms. Das Verfahren terminiert in einem Schritt 210.

Wie in Fig.2 angesprochen, wird in Fig.3 der Schritt 208 aus Fig.2 eingehend erläutert.

Die Verhandlung des Rechners CL1 mit dem Server SV erfolgt derart, daß der Rechner CL1 von dem Server SV eine Liste aller verfügbaren plattformabhängigen Realisierungen der Methode b anfordert (siehe Schritt 301). Die Liste wird von dem Server SV an den Rechner CL1 übertragen. In einem Schritt 302 wählt der Rechner CL1 eine für sein Betriebssystem S1 geeignete Realisierung der Methode b aus, teilt dies dem Server SV mit und fordert eine Übertragung der entsprechenden plattformabhängigen Realisierung der Methode b (siehe Block 102 in Fig.1) in einem Schritt 303 an.

Patentansprüche

1. Verfahren zum Laden einer Funktion, die von einem ersten Rechner (Server) bereitgestellt wird, auf einen zweiten Rechner (Client),
wobei der erste Rechner und der zweite Rechner über ein Netz verbunden sind, in folgenden Schritten:
a) von dem zweiten Rechner wird über das Netz bei dem ersten Rechner angefragt, ob die Funktion in einem plattformabhängigen Code verfügbar ist;
b) falls ein plattformabhängiger Code nicht verfügbar ist, wird ein plattformunabhängiger Code der Funktion von dem ersten Rechner geladen und das Verfahren beendet;
c) ansonsten (der plattformabhängige Code ist verfügbar) werden folgende Schritte durchgeführt:
- es wird überprüft, ob der plattformabhängige Code der Funktion auf dem zweiten Rechner vorhanden ist und, wenn dies der Fall ist, wird der plattformunabhängige Code geladen und das Verfahren beendet;
- falls der plattformabhängige Code auf dem zweiten Rechner nicht vorhanden ist, wird mittels einer Eingabe entschieden, ob der plattformabhängige Code vom ersten Rechner geladen werden soll:
  - wird mittels der Eingabe entschieden, daß der plattformabhängige Code nicht geladen werden soll, wird der plattformunabhängiger Code geladen und das Verfahren beendet;
  - wird mittels der Eingabe entschieden, daß der plattformabhängige Code geladen werden soll, so wird nach einer Verhandlung zwischen dem ersten Rechner und dem zweiten Rechner der plattformabhängige Code der Funktion auf den zweiten Rechner geladen.

2. Verfahren nach Anspruch 1,
bei dem die Verhandlung folgende Schritte umfaßt:
a) der zweite Rechner (Client) fordert von dem ersten Rechner (Server) eine Liste aller verfügbarer plattformabhängiger Realisierungen der Funktion an;
b) die Liste wird von dem ersten Rechner an den zweiten Rechner übertragen;
c) der zweite Rechner wählt eine geeignete Realisierung aus, teilt dies dem ersten Rechner mit und fordert eine Übertragung dieser Realisierung der Funktion an.

3. Verfahren nach Anspruch 1 oder 2,
bei dem nach dem letzten Schritt, dem Laden des plattformabhängigen Codes von dem ersten Rechner auf den zweiten Rechner, eine Sicherheitsüberprüfung dieses plattformabhängigen Codes durchgeführt wird.

4. Verfahren nach Anspruch 3,
bei dem die Sicherheitsüberprüfung folgende Maßnahme umfaßt:
bevor der geladene plattformabhängige Code auf dem zweiten Rechner ausgeführt wird, erfolgt eine Integritätsprüfung des Codes, vorzugsweise mittels eines Virensuchprogramms.

5. Verfahren nach einem der vorhergehenden Ansprüche,
bei dem die Eingabe auf eine der folgenden Arten vorgenommen wird:
a) ein Benutzer gibt an, ob er plattformabhängigen Code von dem ersten Rechner laden will;
b) anhand einer Liste, die Rechner einer sicheren Umgebung umfaßt;
c) es wird immer plattformabhängiger Code geladen, wenn der erste Rechner keine sicherheitsrelevanten Daten und/oder Programme enthält.

6. Verfahren nach einem der vorhergehenden Ansprüche, bei dem der plattformunabhängige Code JAVA-Bytecode ist.

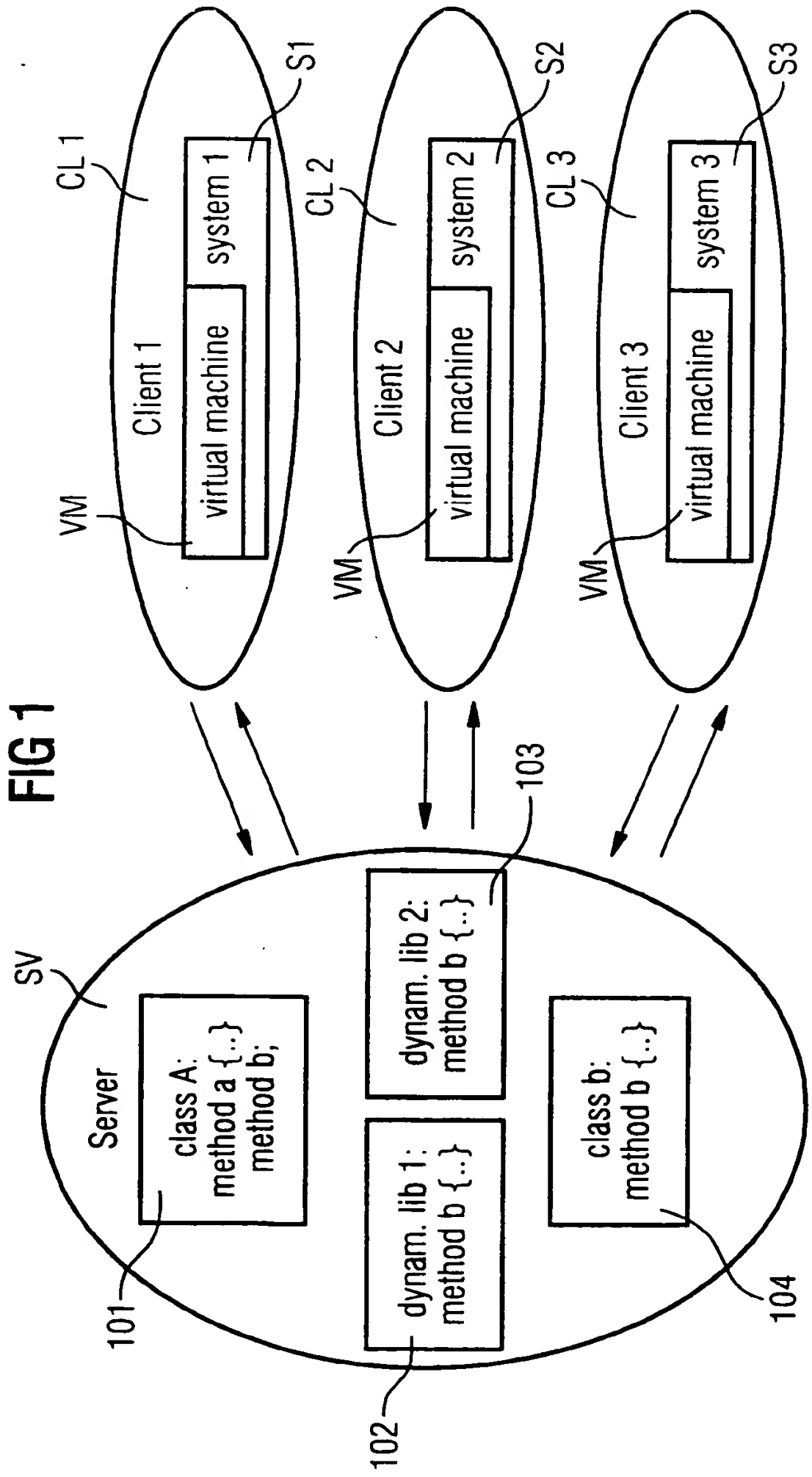

CL 1
CL 2
CL 3
S1
S2
S3
Client 1
Client 2
Client 3
system 1
system 2
system 3
virtual machine
VM
SV
Server
class A:
method a {..}
method b;
dynam. lib 1:
method b {..}
dynam. lib 2:
method b {..}
class b:
method b {..}
101
102
103
104

FIG 1

FIG 2

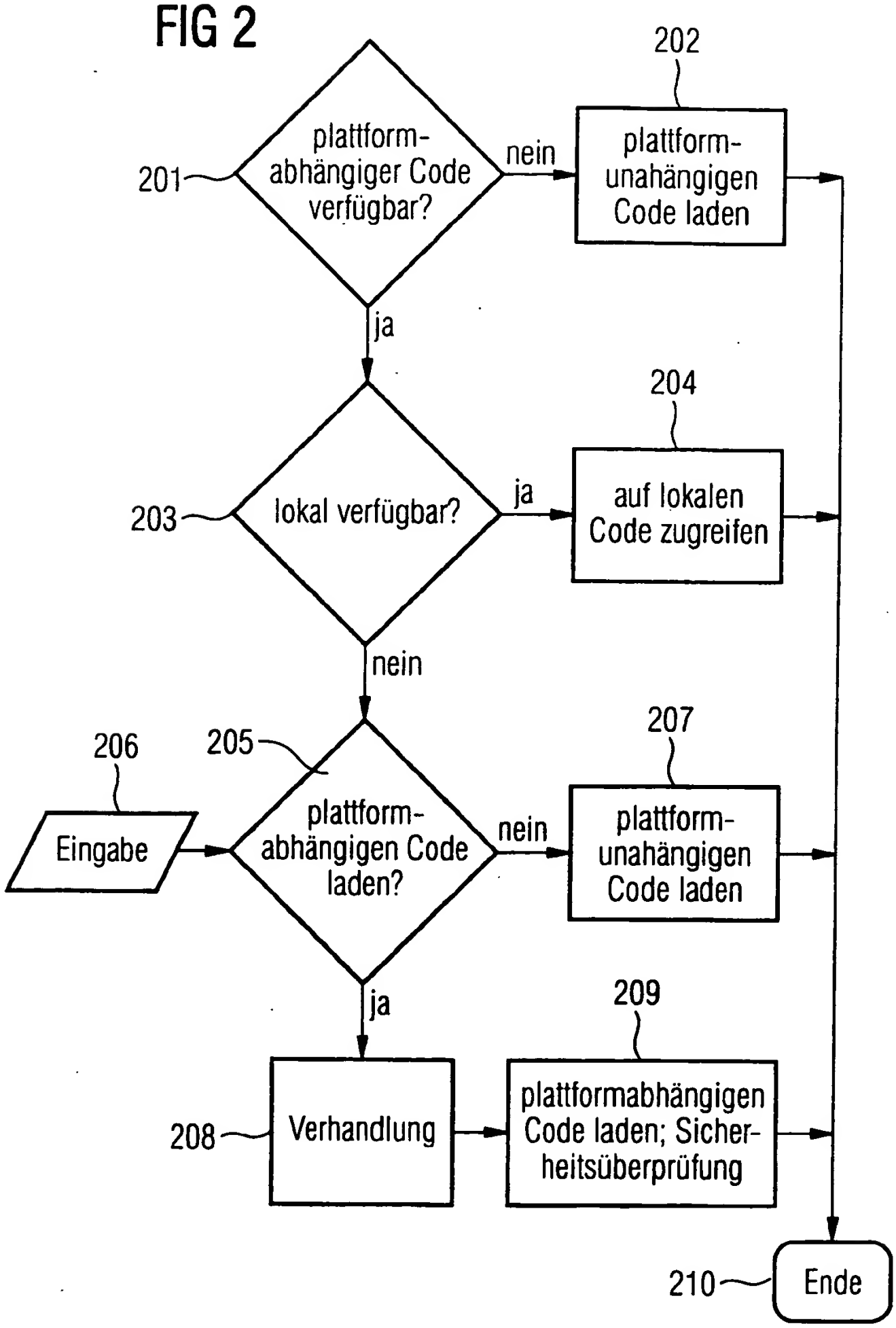
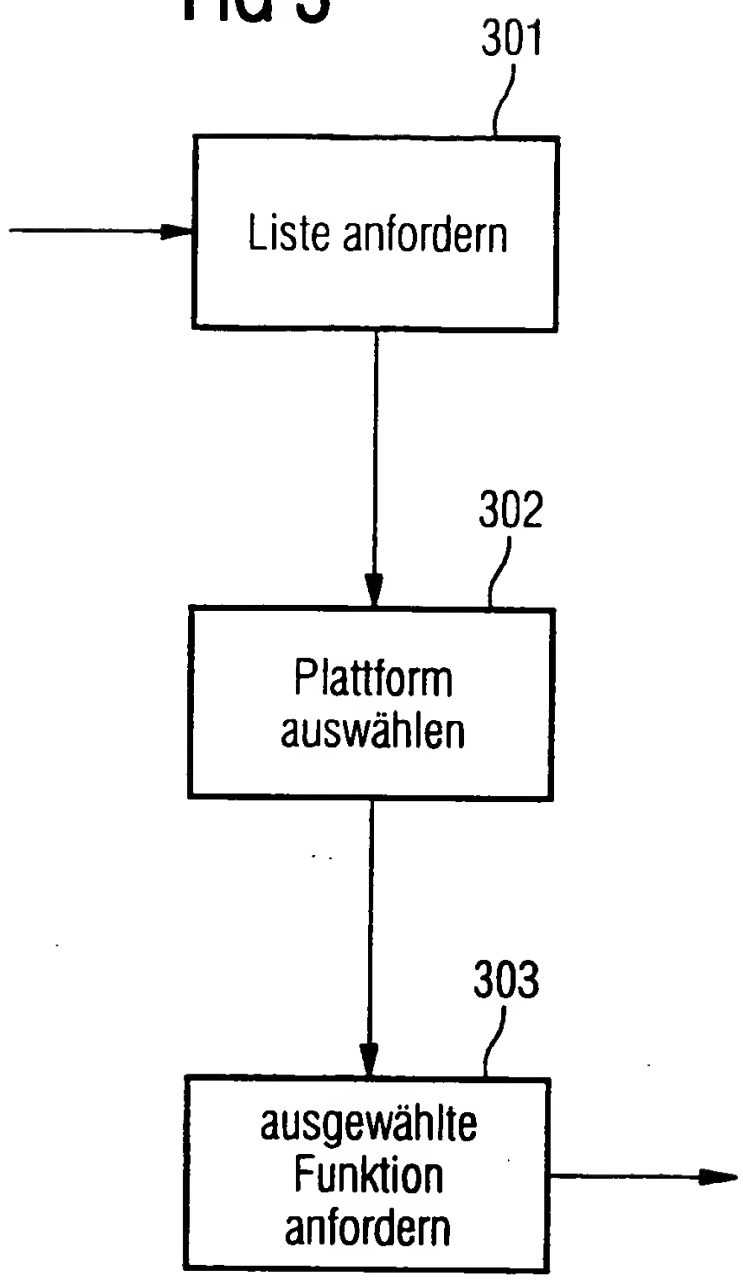

201
plattform-abhängiger Code verfügbar?
nein
202
plattform-unahängigen Code laden
ja
203
lokal verfügbar?
ja
204
auf lokalen Code zugreifen
nein
206
Eingabe
205
plattform-abhängigen Code laden?
nein
207
plattform-unahängigen Code laden
ja
208
Verhandlung
209
plattformabhängigen Code laden; Sicher-heitsüberprüfung
210
Ende

FIG 3


301
Liste anfordern
302
Plattform auswählen
303
ausgewählte Funktion anfordern